Introduction to Model Driven Architecture (MDA)

NCICB Software Development Processes
Facilitating Systems Interoperability

Sashi Thangaraj (SAIC)

Agenda

- What is the MDA?
  - MDA Overview
  - MDA Principles
- Why Model?
- MDA and caBIG
- MDA and Software Development
- MDA Approach
- Case Study – caBIO and MDA
- Q&A

What is the MDA?

- Model Driven Architecture (MDA) is an emerging set of standards and technologies focused on a particular software development style
- The MDA provides a conceptual framework and set of standards
  - Express models
  - Model relationships,
  - Model-to-model transformations
- MDA is based on the
  - Meta-Object Facility (MOF)
  - Unified Modeling Language (UML)
  - XML Metadata Interchange (XMI)
  - Common Warehouse Meta-Model (CWM) modeling specifications
- MDA established by Object Management Group (OMG), a non-profit consortium of 800+ organizations that produces/maintains computer industry specifications for interoperable enterprise applications

> **"MDA works as a reasonable step up from today's popular development techniques."**
>
> — *Grady Booch*

Principles of MDA

- Four principles underlie the OMG’s view of MDA:
  - Well-defined notation models are cornerstone to understanding systems
  - Building systems can be organized around a set of models which are organized into an architectural framework of layers and transformations
  - A formal underpinning for describing models in a set of meta-models facilitates meaningful integration and transformation among models, and is the basis for automation through tools
  - Acceptance and broad adoption of this model-based approach requires industry standards to provide openness to consumers, and foster competition among vendors

Why Model?

- All forms of engineering rely on models to understand complex, real-world systems
- Models facilitate the communication of key system characteristics and complexities to various stakeholders
- Models provide abstractions of a physical system that allow engineers to reason about the system by ignoring extraneous details while focusing on relevant ones
- Models are used to reason about specific properties of the system when aspects of the system change and can assist in predicting system qualities
- Depending on the context, different elements can be modeled which provide different views which ultimately facilitates:
  - analyzing problems
  - proposing solutions
- Applying different kinds of models provides a well-defined style of development, providing ability to re-use common approaches

MDA and caBIG

- The use of MDA will facilitate interoperability between caBIG systems
  - Interoperability is key for data sharing in federated systems
- MDA approaches will communicate key system characteristics to caBIG participants
- caBIG silver/gold compatibility guidelines specify the use of standards based information models for facilitating interoperability
- The caBIG architecture workspace will assist in recommending standard document templates describing MDA artifacts (e.g. use cases)

MDA and Software Development

- Several software development processes leverage MDA to varying degrees:
  - Rational Unified Process (RUP)
  - Extreme Programming (XP)
  - Agile Programming
  - Home Grown Process
  - Combinations
    - RUP and XP
  - Others
- In each software development process, there are different ways of developing software
  - Code only
  - Model only
  - Model is our code -> Code is our model
- Software development tools and technologies can assist in developing software based on MDAs making it practical and efficient to apply

MDA Approach

- Analyze the problem space and develop the artifacts for each scenario
  - Use Cases
- Design the system by developing artifacts based on the use case
  - Class Diagram
  - Sequence Diagram
- Use meta-model tools to generate the code

Case Study – caBIO and MDA

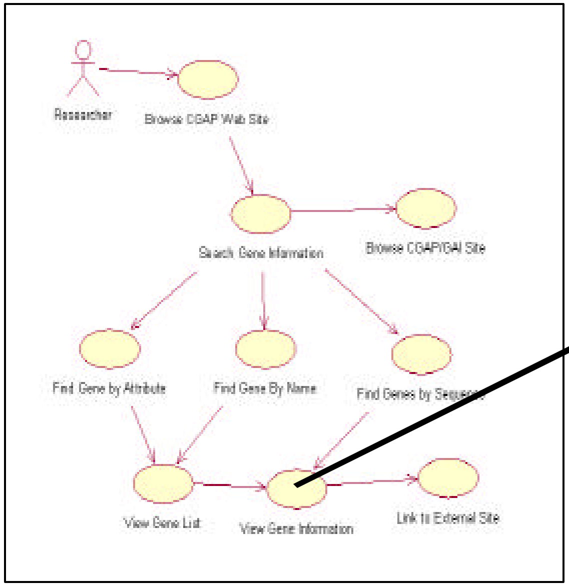
- caBIO Overview
- caBIO Problem Statement
  - Use Cases
- caBIO Design Artifacts
  - UML Class Diagrams
  - Sequence Diagrams
  - Architecture
- caBIO Software Development
  - Code Generation Tools
  - APIs
- caBIO Testing and Deployment

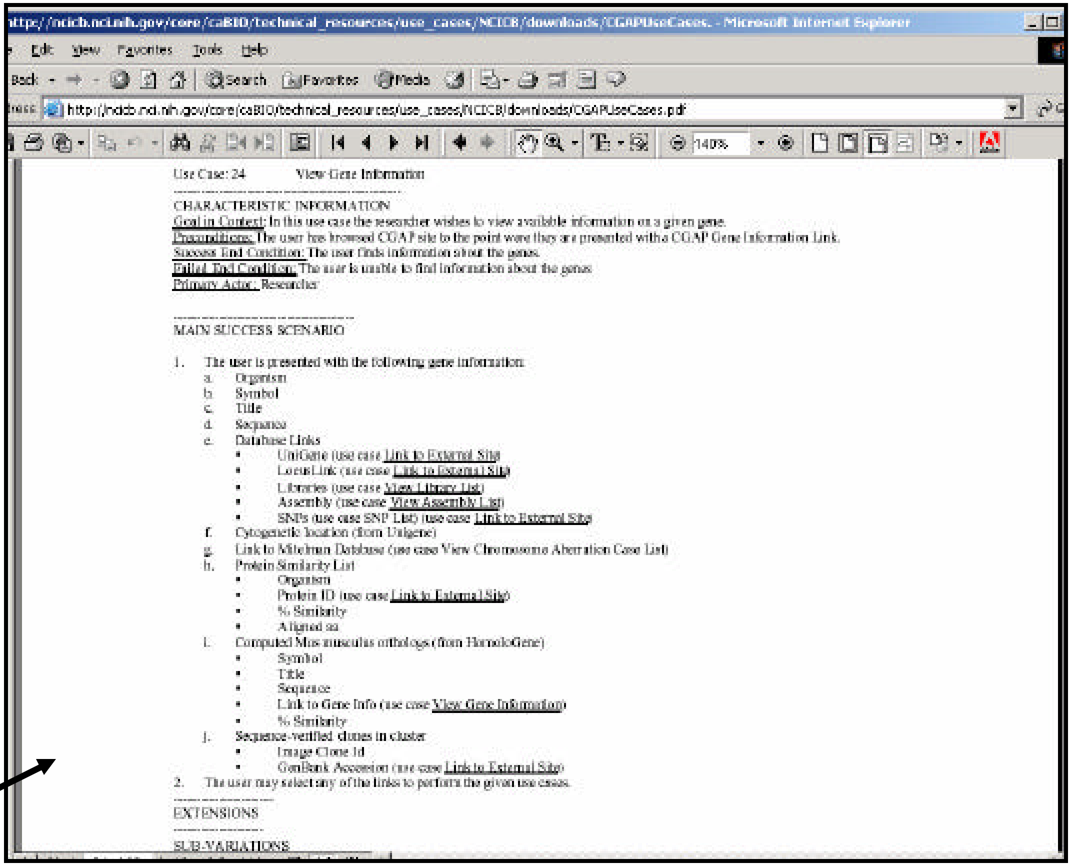
caBIO Overview

- The cancer Bioinformatics Infrastructure Objects (caBIO) is a service-oriented based infrastructure supporting multi-disciplinary scientific research studies
- caBIO provides standard object models and uniform API (Java, SOAP, HTTP-XML, Perl) access to a variety of intramural and extramural genomic, biological, and clinical data sources
- caBIO objects simulate the behavior of actual biomedical components such as genes, sequences, chromosomes, sequences, cellular pathways, ontologies, clinical protocols, etc.
- caBIO is “open source” and provides an abstraction layer that allows developers to access genomic information using a standardized tool set without concerns for implementation details and data management

caBIO Problem Space – Use Cases

- Description
- Actors
- Basic Course
- Alternative Course




ttp://ncicb.nci.nih.gov/core/caBIO/technical_resources/use_cases/NCICB/downloads/CGAPUseCases. - Microsoft Internet Explorer
Edit View Favorites Tools Help
Back Search Favorites Media
http://ncicb.nci.nih.gov/core/caBIO/technical_resources/use_cases/NCICB/downloads/CGAPUseCases.pdf
140%

Use Case: 24 View Gene Information

CHARACTERISTIC INFORMATION

Goal in Context: In this use case the researcher wishes to view available information on a given gene.
Preconditions: The user has browsed CGAP site to the point were they are presented with a CGAP Gene Information Link.
Success End Condition: The user finds information about the genes.
Failed End Condition: The user is unable to find information about the genes.
Primary Actor: Researcher

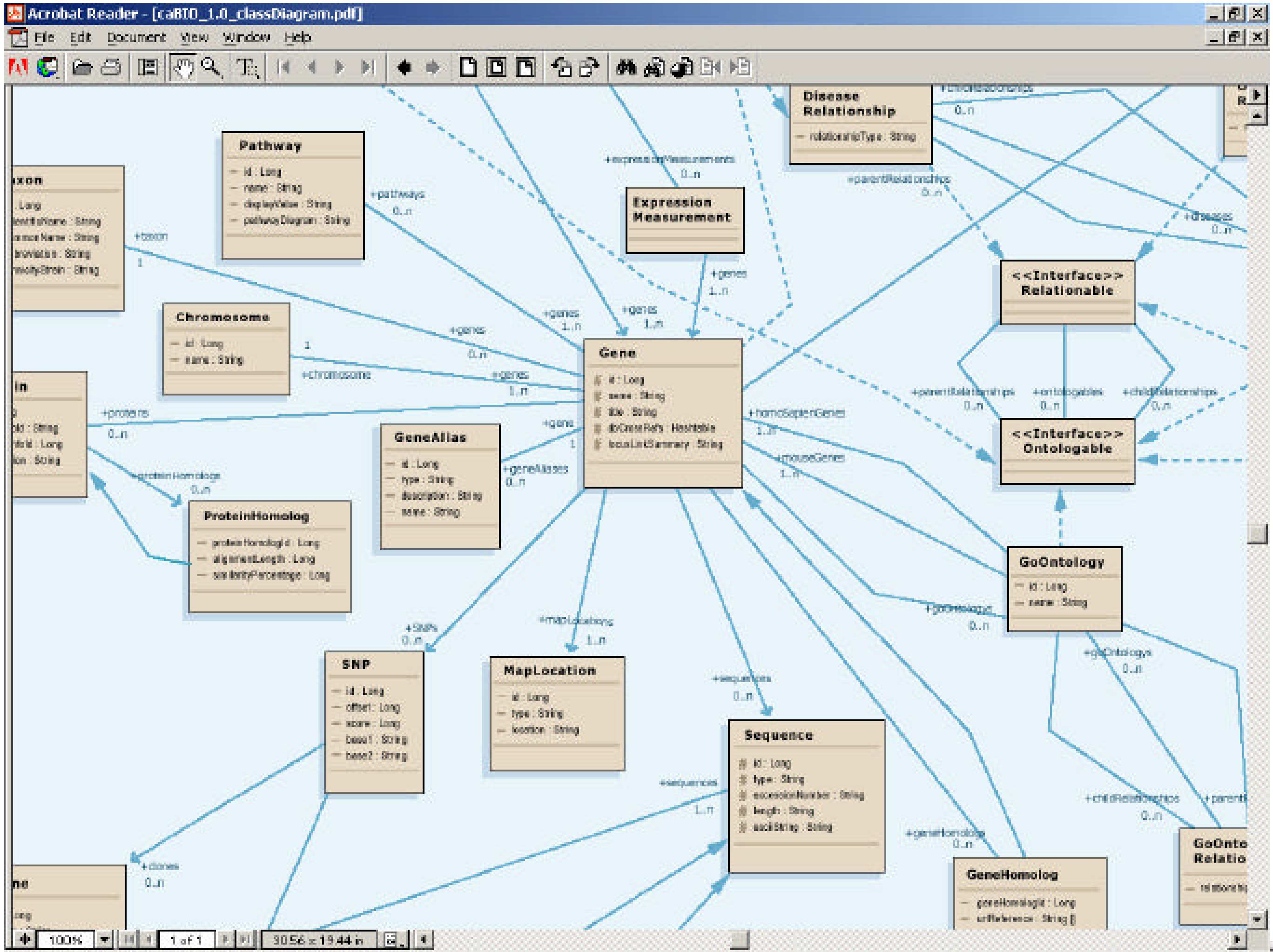
MAIN SUCCESS SCENARIO

1. The user is presented with the following gene information:
   a. Organism
   b. Symbol
   c. Title
   d. Sequence
   e. Database Links
      - UniGene (use case Link to External Site)
      - LocusLink (use case Link to External Site)
      - Libraries (use case View Library List)
      - Assembly (use case View Assembly List)
      - SNPs (use case SNP List) (use case Link to External Site)
   f. Cytogenetic location (from Unigene)
   g. Link to Mitelman Database (use case View Chromosome Aberration Case List)
   h. Protein Similarity List
      - Organism
      - Protein ID (use case Link to External Site)
      - % Similarity
      - Aligned aa
   i. Computed Mus musculus orthologs (from HomoloGene)
      - Symbol
      - Title
      - Sequence
      - Link to Gene Info (use case View Gene Information)
      - % Similarity
   j. Sequence-verified clones in cluster
      - Image Clone Id
      - GenBank Accession (use case Link to External Site)
2. The user may select any of the links to perform the given use cases.

EXTENSIONS

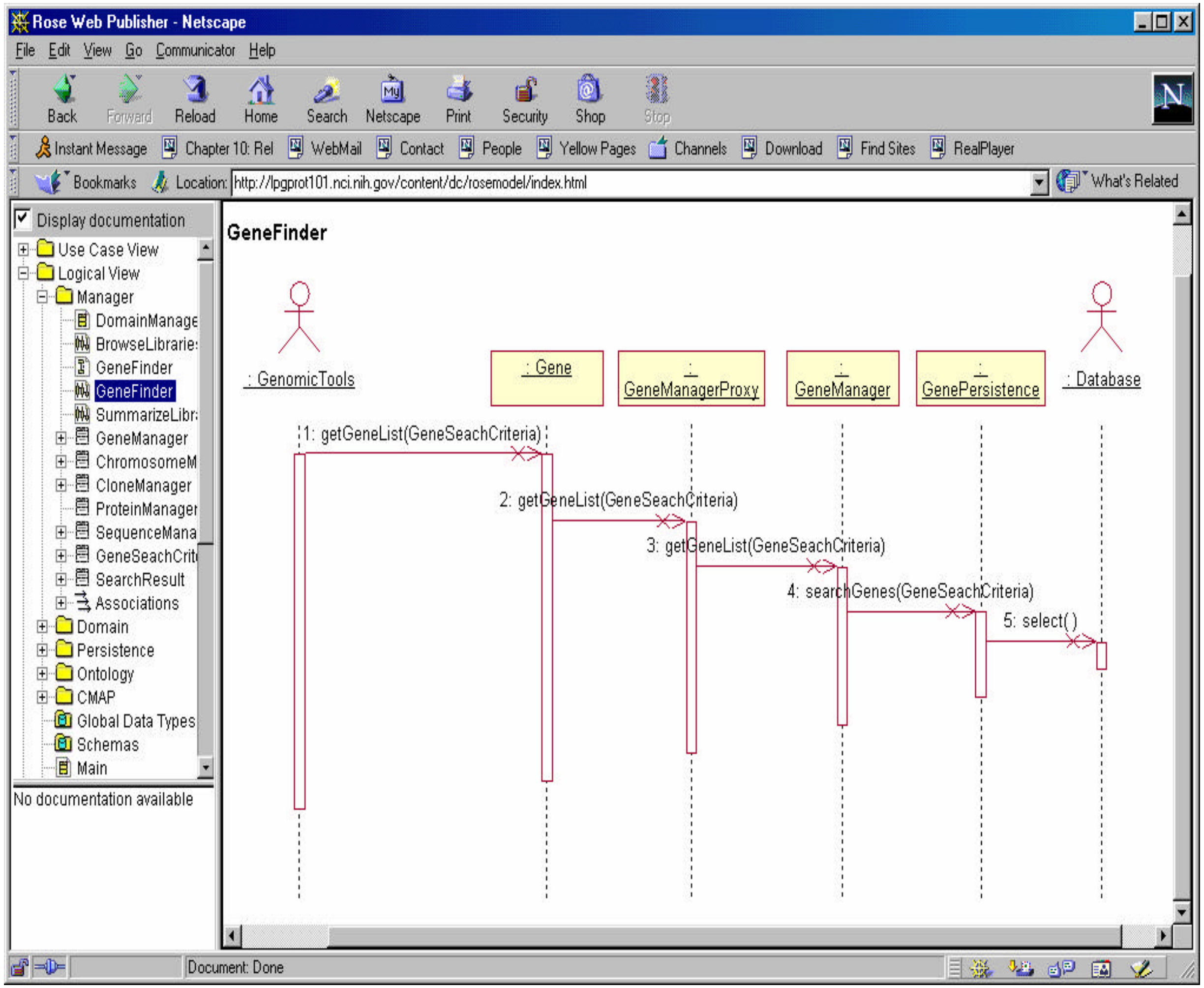
SUB-VARIATIONS

caBIO Design Artifacts – UML Class Diagrams

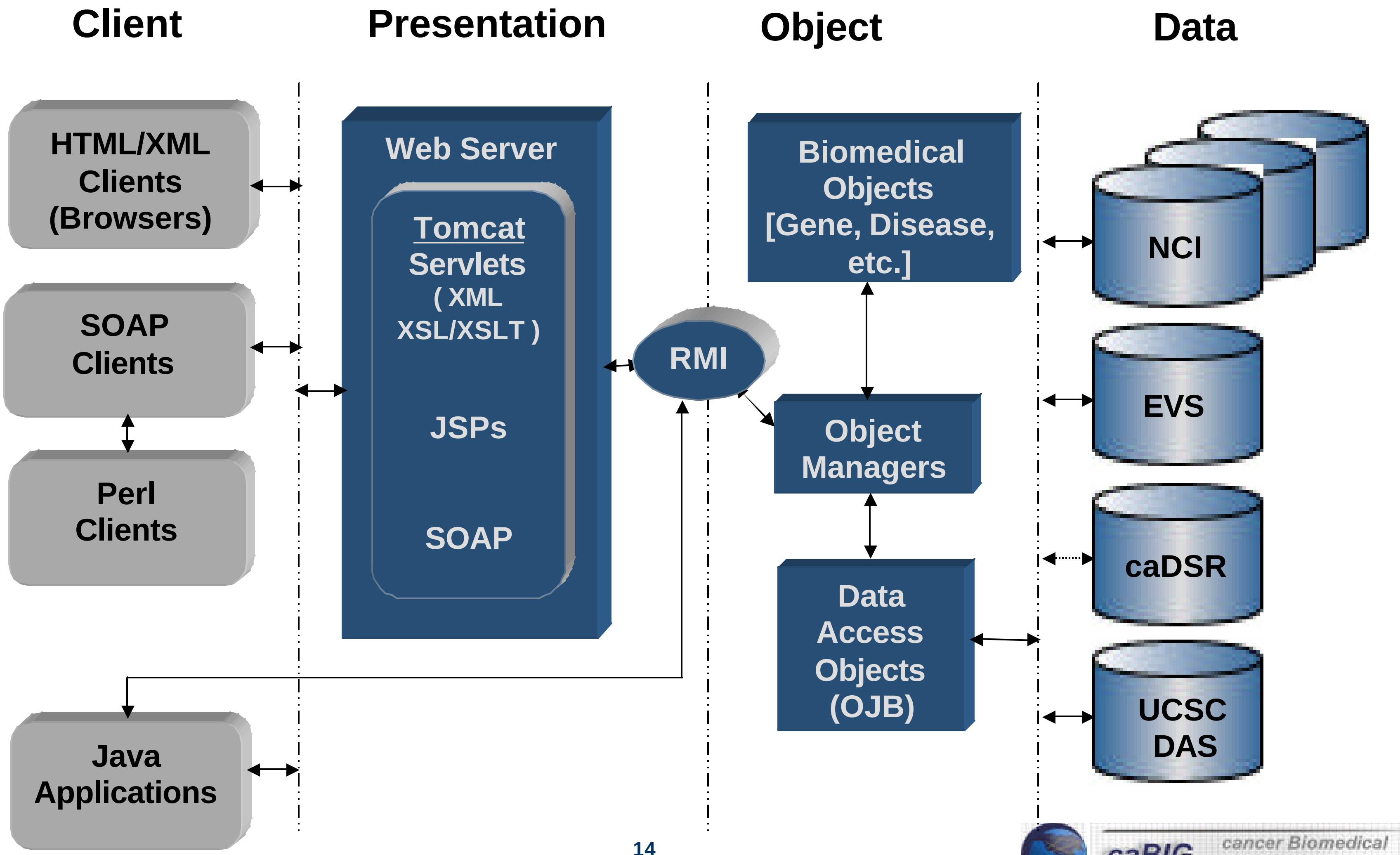


caBIO Design Artifacts – State Transition Diagrams

- Sequence diagrams model the flow of logic within your system visually, enabling validation and documentation of logic


Rose Web Publisher - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Instant Message Chapter 10: Rel WebMail Contact People Yellow Pages Channels Download Find Sites RealPlayer
Bookmarks Location: http://lpgprot101.nci.nih.gov/content/dc/rosemodel/index.html
What's Related
☑ Display documentation
Use Case View
Logical View
Manager
DomainManage
BrowseLibrarie
GeneFinder
GeneFinder
SummarizeLibr
GeneManager
ChromosomeM
CloneManager
ProteinManager
SequenceMana
GeneSeachCrit
SearchResult
Associations
Domain
Persistence
Ontology
CMAP
Global Data Types
Schemas
Main
No documentation available
GeneFinder
: GenomicTools
: Gene
: GeneManagerProxy
: GeneManager
: GenePersistence
: Database
1: getGeneList(GeneSeachCriteria)
2: getGeneList(GeneSeachCriteria)
3: getGeneList(GeneSeachCriteria)
4: searchGenes(GeneSeachCriteria)
5: select( )
Document: Done

caBIO Design Artifacts - Architecture

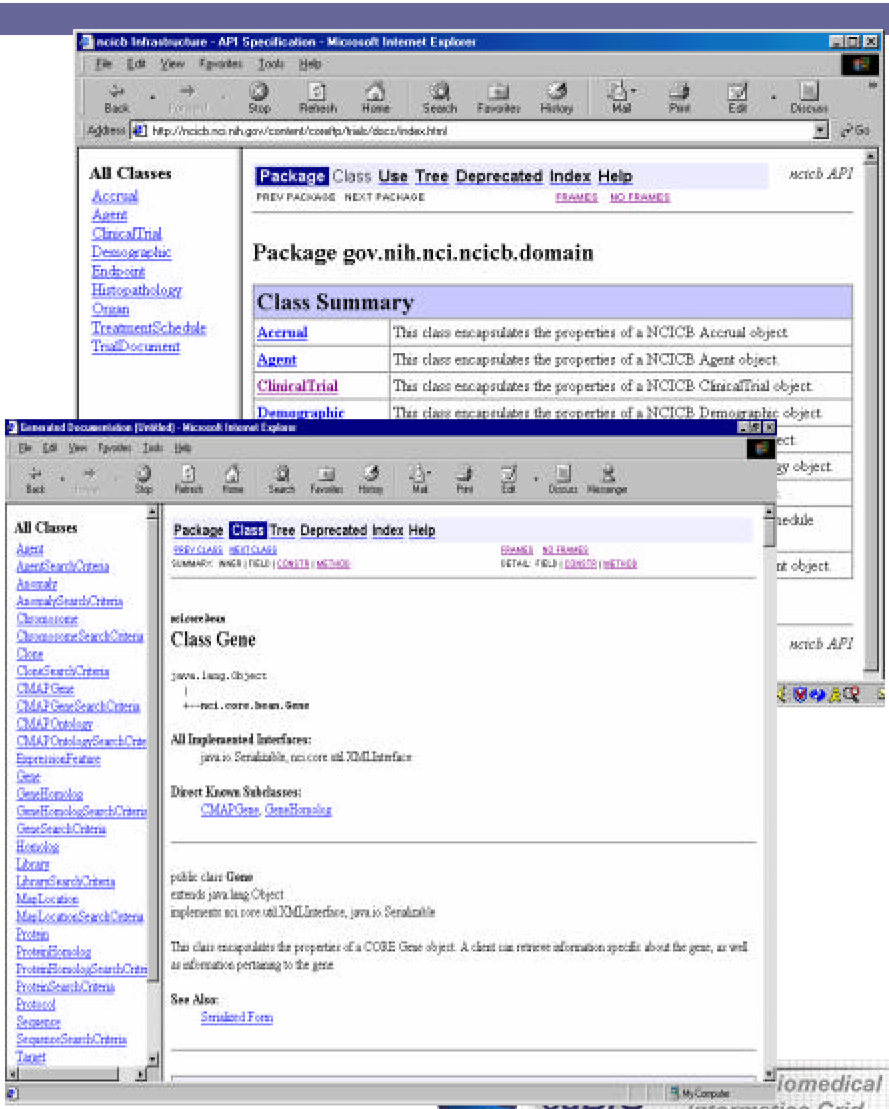

Client
Presentation
Object
Data
HTML/XML Clients (Browsers)
SOAP Clients
Perl Clients
Java Applications
Web Server
Tomcat Servlets (XML XSL/XSLT)
JSPs
SOAP
RMI
Biomedical Objects [Gene, Disease, etc.]
Object Managers
Data Access Objects (OJB)
NCI
EVS
caDSR
UCSC DAS

caBIO Software Development – Code Generation Tools

- caBIO leverages in house code generation tools for generating APIs
- There are a variety of third-party and open source code generation tools for generating Java, SOAP, HTTP, and Perl APIs
- Code generation tools rely on templates that generate code directly from the UML model (XMI file)
- Automatic code generation facilitates ease of maintenance
- Standards based model driven automated design and development processes facilitate ease of maintenance!

caBIO Software Development – APIs

- **Java**
  - Query/retrieve biomedical objects directly via RMI
- **HTTP-XML**
  - Properly formed URLs in any web browser/client can retrieve XML-formatted object data directly
- **SOAP**
  - SOAP client in any language/environment can send request to NCICB server for object data
  - SOAP-XML envelope and payload returned
- **caBIOperl**
  - caBIOperl wraps lower-level SOAP API
  - Shields developers from SOAP calls and XML parsing



ncicb Infrastructure - API Specification - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address http://ncicb.nci.nih.gov/content/coreftp/trials/docs/index.html
Go

All Classes
Accrual
Agent
ClinicalTrial
Demographic
Endpoint
Histopathology
Organ
TreatmentSchedule
TrialDocument

Package Class Use Tree Deprecated Index Help — ncicb API
PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

Package gov.nih.nci.ncicb.domain

Class Summary	
Accrual	This class encapsulates the properties of a NCICB Accrual object.
Agent	This class encapsulates the properties of a NCICB Agent object.
ClinicalTrial	This class encapsulates the properties of a NCICB ClinicalTrial object.
Demographic	This class encapsulates the properties of a NCICB Demographic object.

Generated Documentation (Untitled) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Messenger
My Computer

All Classes
Agent
AgentSearchCriteria
Anomaly
AnomalySearchCriteria
Chromosome
ChromosomeSearchCriteria
Clone
CloneSearchCriteria
CMAPGene
CMAPGeneSearchCriteria
CMAPOntology
CMAPOntologySearchCrite
ExpressionFeature
Gene
GeneHomolog
GeneHomologSearchCriteria
GeneSearchCriteria
Homolog
Library
LibrarySearchCriteria
MapLocation
MapLocationSearchCriteria
Protein
ProteinHomolog
ProteinHomologSearchCrite
ProteinSearchCriteria
Protocol
Sequence
SequenceSearchCriteria
Target

Package Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

nci.core.bean
Class Gene

```
java.lang.Object
  |
  +--nci.core.bean.Gene
```

All Implemented Interfaces:
java.io.Serializable, nci.core.util.XMLInterface

Direct Known Subclasses:
CMAPGene, GeneHomolog

public class **Gene**
extends java.lang.Object
implements nci.core.util.XMLInterface, java.io.Serializable

This class encapsulates the properties of a CORE Gene object. A client can retrieve information specific about the gene, as well as information pertaining to the gene.

See Also:
Serialized Form

caBIO Testing and Deployment

- Testing occurs in various stages:
  - Development (Unit) Testing
  - Integration Testing
  - System Testing
  - Production Testing
- Test cases are created for each use case
- Test scripts are created to test all test cases and APIs
- Data validation is an important component of testing
- caBIO is deployed to each test server and the production server via standard build processes
  - Apache ANT, an open source Java based build tool, is leveraged
- All MDA artifacts and artifact versions are maintained under Configuration Management (CM) control
  - Concurrent Versioning System (CVS), an open source CM tool, is leveraged

Q & A

- http://ncicb.nci.nih.gov/core/caBIO